STRUTS 2 TYPE CONVERSION

Everything on a HTTP request is treated as a **String** by the protocol. This includes numbers, booleans, integers, dates, decimals and everything else. Every thing is a string according to HTTP. However, in the Struts class, you could have properties of any data types. Howe does Struts autowire the properties for you ?

Struts uses a variety of type converters under the covers to do the heavy lifting. For example, if you have an integer attribute in your Action class, Struts automatically converts the request parameter to the integer attribute without you doing anything. By default, Struts comes with a number of type converters. Some of them are listed below and if you are using any of them then you have nothing to worry about:

- Integer, Float, Double, Decimal
- Date and Datetime
- Arrays and Collections
- Enumerations
- Boolean
- BigDecimal

Some times when you are using your own data type, it is necessary to add your own converters to make Struts aware how to convert those values before displaying. Consider the following POJO class **Environment.java**.

```
package com.tutorialspoint.struts2;

public class Environment {
   private String name;
   public  Environment(String name)
   {
      this.name = name;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

This is a very simple class that has an attribute called **name**, so nothing special about this class. Let us create another class that contains information about the system - **SystemDetails.java**. For the purpose of this exercices, I have hardcoded the Environment to "Development" and the Operating System to "Windows XP SP3". In a real project, you would get this information from the system configuration. So let us have following action class:

```
package com.tutorialspoint.struts2;
import com.opensymphony.xwork2.ActionSupport;

public class SystemDetails extends ActionSupport {
   private Environment environment = new Environment("Development");
   private String operatingSystem = "Windows XP SP3";

   public String execute()
   {
      return SUCCESS;
   }
   public Environment getEnvironment() {
```

```
      return environment;
   }
   public void setEnvironment(Environment environment) {
      this.environment = environment;
   }
   public String getOperatingSystem() {
      return operatingSystem;
   }
   public void setOperatingSystem(String operatingSystem) {
      this.operatingSystem = operatingSystem;
   }
}
```

Next let us create a simple JSP file **System.jsp** to display the Environment and the Operating System information.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>System Details</title>
</head>
<body>
   Environment: <s:property value="environment"/><br/>
   Operating System:<s:property value="operatingSystem"/>
</body>
</html>
```

Let us wire the **system.jsp** and the **SystemDetails.java** class together using **struts.xml**. The SystemDetails class has a simple execute() method that returns the string "SUCCESS".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <constant name="struts.devMode" value="true" />
   <package name="helloworld" extends="struts-default">
      <action name="system"

            method="execute">
         <result name="success">/System.jsp</result>
      </action>
   </package>
</struts>
```

Right click on the project name and click **Export > WAR File** to create a War file. Then deploy this WAR in the Tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/HelloWorldStruts2/system.action. This will give you following screen:

What is wrong with the above output? Struts knows how to display and convert the string "Windows XP SP3" and other built-in data types, but it does not know what to do with the the property of **Environment** type. So, it simply called the **toString()** method on the class. To resolve this problem, let us now create and register a simple **TypeConverter** for the Environment class. Create a class called **EnvironmentConverter.java** with the following.

```
package com.tutorialspoint.struts2;

import java.util.Map;
import org.apache.struts2.util.StrutsTypeConverter;
```

```
   public class EnvironmentConverter extends StrutsTypeConverter {
      @Override
      public Object convertFromString(Map context, String[] values,
                                      Class clazz) {
      Environment env = new Environment(values[0]);
      return env;
   }

   @Override
   public String convertToString(Map context, Object value) {
      Environment env  = (Environment) value;
      return env == null ? null : env.getName();
   }
}
```

The **EnvironmentConverter** extends the **StrutsTypeConverter** class and tells Struts how to convert Environment to a String and vice versa by overriding two methods **convertFromString()** and **convertToString()**. Let us now register this converter before we us it in our application. There are two ways to register a converter. If the converter will be used only in a particular action, then you would have to create a property file needs to be named as **'[action-class]'-converstion.properties**, So, in our case we create a file called **SystemDetails-converstion.properties** with the following registration entery:

```
environment=com.tutorialspoint.struts2.EnvironmentConverter
```

In the above example, "environment" is the name of the property in the **SystemDetails.java** class and we are telling Struts to use the **EnvironmentConverter** for converting to and from this property. However, we are not going to do this, Instead we are going to register this converter globally so that it can be used throughout the application. To do this, create a property file called **xwork-conversion.properties** in the **WEB-INF/classes** folder with the following line:

```
com.tutorialspoint.struts2.Environment = \
            com.tutorialspoint.struts2.EnvironmentConverter
```

This simply registers the converter globally, so that Struts can automatically do the conversion every time it encounters an object of type **Environment**. Now, if you re-compile and re-run the program, you will get a better output as follows:

Obviously, now result is better which means our Struts convertor is working fine. This is how you can create multiple convertors and register them to use as per your requirements.